***Описание работы по программной части***  
Что здесь будет:

1. Подготовка виртуальной машины к программам
2. Установка программ
3. Настройка программ
4. Первые шаги к интеграции
5. Периодически будет обновляться (по мере заполнения промежуточных версий на большом компе)

На виртуалке:

1. Стандартный набор для VM – см. туториал из веб-разработки
2. Virtualenv, PyCharm Community Edition (5.0.4), Java 8, Ubuntu 14/04 LTS x64
3. Ниже есть список программ для установки и инструкции по их установке (тестим на Debug машине, потом ставим на Release, сюда пишем туториалы, как правильно и не криво ставить)

На винде:

1. DS9
2. PyCharm, мб Spark
3. GitHub xterm

***Полезные ссылки для работы***

1. <http://hilite.me/> - для подсветки кода и вставки сюда – удобно (+ внешние рамки – вообще норм)
2. <https://github.com/sergotail> - мой GitHub репозиторий
3. <http://spark.apache.org/streaming/> - Spark Streaming, возможно, пригодится для организации потоковой обработки данных

***Инструкция по установке (и настройке, а также интеграции в будущем) программ для обработки астрономических изображений.***

Список программ для работы с астрономическими изображениями:

1. IRAF – аналог SEXtractor, вроде как поддерживает вызов функции через API (вроде как на C++, это выяснится в ходе работы)
2. Aladin Sky Atlas – программа для визуализации, должна сильно помогать в понимании устройства астрономических изображений, нужна Desktop версия (работает на Java, так что требуется Java)
3. Возможно, программа, аналогичная Total Commander
4. SExtractor – тут важна как монолиттная програма, так и в виде библиотеки
5. Astrometry.net – разобраться, как пользоваться, и вроде как есть исходники, надо с ними разобраться, чтобы понять, как использовать их математику для каталогизации объектов  
   (<http://astrometry.net/use.html>, <http://astrometry.net/doc/readme.html>, <http://astrometry.net/doc/>)
6. **Подготовка виртуальной машины к программам**
   1. Создание общих папок. Это нужно для того, чтобы изображения хранить в винде и открывать их там с помощью DS9, но при этом чтобы можно было их обрабатывать с помощью программ на виртуалке. Тут <http://aboutubuntu.ru/content/nastroika-virtualnoi-mashiny-virtualbox-s-gostevoi-ubuntu-na-windows-khoste> описан процесс. Я буду создавать такие папки для каждой из виртуалок отдельно. В винде: Документы – VirtualBox – Shared Folders. Пример для папки AstroKiraDebug:
      1. Устройства (окно в виртуальной машине VirtualBox) – Общие папки  
         Галочки «постоянная папка» и «авто подключение» - для работы постоянно и после перезапуска машины тоже
      2. Перезагружаем машину, в паке /media появится папка sf\_AstroKiraDebug
      3. Добавляемся в группу vboxsf: sudo adduser ser vboxsf
      4. Перелогиниваемся, теперь имеем доступ к папке, на всякий случай ставим к ней полный доступ: sudo chmod 777 sf\_AstroKraDebug/
      5. Создаём symlinnk в удобное место на папку:  
         sudo ln -s /media/sf\_AstroKraDebug/ ~/SharedFolder
7. **Установка IRAF**
   1. wget <ftp://iraf.noao.edu/iraf/v216/PCIX/iraf.lnux.x86_64.tar.gz>
   2. ~/Dev/Astronomy/iraf/ - тут я создал домашнюю директорию, сюда распаковать:  
      tar –zxf iraf.lnux.x86\_64.tar.gz
   3. Есть 2 вида установки: обычная (1 юзер) и системная (для всех), лучше ставить системную, т.к. может в будущем понадобиться работа с папками, для которых нужны root права: sudo ./install –system
   4. Во время установки задать папки для хранения изображений и кэша, я задал такие:

set home = "/home/ser/iraf/"  
set imdir = "/home/ser/Dev/Astronomy/iraf.tmp/images/ser/"  
set cache = "/home/ser/Dev/Astronomy/iraf.tmp/cache/ser/"

Возможно, потом будет удобнее поменять папку для хранения изображений на общую (см. выше SharedFolder)

* 1. Затем создаём login.cl для своего юзера, это делается в домашней папке, поэтому сначала mkdir ~/iraf. Затем: mkiraf
  2. Далее идёт настройка IRAF, по сути, тут <http://www.astronet.ru/db/msg/1216409> описан туториал по работе с IRAF. Что точно понадобится потом:
     1. Вызов функций IRAF из кода другой программы и/или
     2. Интеграция с другой системой (например, на спарке)

1. **Установка Aladin Sky Atlas**
   1. 2 варианта запуска: Java Web Start: <http://aladin.u-strasbg.fr/java/download/aladin.jnlp> или shell launcher:  
      wget <http://aladin.u-strasbg.fr/java/download/Aladin.tar>  
      Распаковал в ~/Dev/Astronomy (там папка уже общая есть в архиве)
   2. Запуск: ./Aladin из папки Aladin
2. Установка SExtractor
   1. Есть два вида этой программы: монолитная и SEP: <https://sep.readthedocs.org/en/v0.5.x/> - тут описано, как скачать, установить и т.д. ВАЖНО: для установки SEP используется pip, при установке Jupyter (см.ниже) его устанавливаем

***Инструкции по установке Spark и всего сопутствующего***

Предполагается наличие виртуалки, см. инструкцию из курса по веб-разработке.  
Сначала будет установлен только Spark, с ним буду работать через IPython, для этого буду использовать ещё Anaconda, пока что БЕЗ ВИРТУАЛЬНОГО ОКРУЖЕНИЯ.  
Как устроено: Anaconda обеспечивает python и его библиотеками, Spark – вычислительное средство, Jupyter IPython Notebook – для написания скриптов для Spark-а через веб-интерфейс. Kira – надстройка над Spark, предоставляет API SExtractor-а для работы с FITS изображениями.

1. **Установка Anaconda**
   1. wget URL, взять с сайта последнюю версию
   2. Добвить в .bashrc  
      export PATH="/home/ser/anaconda3/bin:$PATH"
2. **Установка Jupiter**
   1. Можно создать отдельное окружение (виртуальное), но мы не делаем этого, ибо нет необходимости
   2. Установка Jupyter: conda install jupyter
   3. Запуск: jupyter notebook

TODO: Возможно, в будущем пригодится: <https://github.com/apache/incubator-toree> - для взаимодействия со Spark-ом

1. **Установка Spark**
   1. Качаем пакет с Hadoop, я распаковал в Dev/Spark/
   2. Добвить в .bashrc путь, заодно задать SPARK\_HOME

export SPARK\_HOME="/home/ser/Dev/Spark/spark-1.6.0-bin-hadoop2.6"  
export PATH="$SPARK\_HOME/bin:$PATH"

* 1. Для связывания Спарка и Юпитера можно использовать пакет findspark, скачаем его с помощью pip
     1. Установка pip   
        sudo apt-get install python-pip
     2. Установка findspark  
        pip install findspark
  2. Теперь нужно проверить, что всё работает
     1. Проверка работоспособности Spark  
        pyspark  
        quit() – примечание: ВАЖНО! Не exit()!!!
     2. Запуск: jupyter notebook – в веб-интерфейсе создаём новый Notebook, в нём пишем код, шаблон для скриптов приведён ниже:

Код, который в Jupyter IPython Notebook использует findspark для поиска Spark-а (локальный режим) + пара примеров (для конфигурации, описанной в данном документе, в зависимости от версий менять пути к файлам или версии программ в коде):

# import main libraries

import findspark

import os

# configure environment

os.environ["SPARK\_HOME"] = "/home/ser/Dev/Spark/spark-1.6.0-bin-hadoop2.6"

os.environ["PYSPARK\_PYTHON"] = "/home/ser/anaconda3/bin/python"

os.environ['PYSPARK\_SUBMIT\_ARGS'] = "--master local[4] pyspark-shell"

findspark.init()

# then we can import spark modules (and any other of course)

# import numpy as np

# import sys

# import sep

# from operator import add

import pyspark

from pyspark import SparkContext

# main (example)

if \_\_name\_\_ == "\_\_main\_\_":

sc = SparkContext(appName="SourceExtractor")

rdd = sc.fitsData("/home/ser/Dev/Kira/Kira-master/scratch/spark/data/")

catalog = rdd.map(lambda (key, fits): (key, extract(np.copy(fits))))

catalog.saveAsTextFile("temp-output")

sc.stop()

# other example

#sc = pyspark.SparkContext("local", "Simple App")

#logData = sc.textFile("/home/ser/test-start.py").cache()

#numAs = logData.filter(lambda s: 'a' in s).count()

#numBs = logData.filter(lambda s: 'b' in s).count()

#print("Lines with a: %i, with b: %i" % (numAs, numBs))

# one more example

#lst = sc.parallelize(range(1000))

#lst.collect()

1. **Установка Kira.**
2. <https://github.com/BIDS/Kira> - репозиторий Kira, может быть полезно (если будут обновления)
3. wget <https://github.com/BIDS/Kira/archive/master.zip>
4. Распаковать архив, я распаковал в Dev/Kira
5. Установка Maven, с его помощью Kira собирается:  
   sudo apt-get install maven
6. Если собираем Kira первый раз, то сначала надо установить один jar пакет:  
   mvn install:install-file -Dfile=lib/jfits-0.94.jar -DgroupId="org.esa" -DartifactId="fits" -Dpackaging="jar" -Dversion="0.94"
7. Далее из папки Kira-master:  
   mvn clean package
8. В папке target появится программа
9. ВАЖНО: в папке scratch находится spark версия Kira, собираем её!  
   (при этом выше сборка не факт, что будет нужна, но пусть пока будет)
10. ВАЖНО: либо в SPARK\_HOME указываем путь без bin на конце, либо везде в Kira убираем префикс bin для SPARK\_HOME, пока что выбрал первый путь + в PATH указал $SPARK\_HOME/bin для использования Spark-а отдельно
11. В scala коде есть использование путей к данным в компах авторов, исправить на мои пути к данным (FITS)